



## Article

# Integrating Assessment in a CSCL Macro-Script Authoring Platform

George Chatzimichalis  and Andreas Papasalouros \* 

Department of Mathematics, University of the Aegean, 83200 Karlovassi, Greece

\* Correspondence: andpapas@aegean.gr

**Abstract:** Collaborative learning entails the involvement and the cooperation of a group of persons with the purpose of learning. Collaborative learning scripts aim to orchestrate the complex interaction among group members while Computer Supported Collaborative Learning scripts (CSCL scripts) is the research field in which IT techniques are involved in the management of the aspects of such an interaction. This article presents assessment-related aspects of an existing CSCL script authoring and deployment platform called *COSTLyP*. Assessment, nowadays, is considered as a vital constituent of CSCL scripts since it may affect some of their necessary components and mechanisms. The outcome of the implementation of an assessment plan may determine what should be the next step in a collaboration activity or what actions should be undertaken to bridge the gap between the expected results and the achieved level of knowledge or expertise. At the same time, assessment can also verify the regulation level that is required within each group; consequently, these scripts should be flexibly designed in order to adapt their evolution to the real needs of the participants.

**Keywords:** collaborative learning; CSCL scripts; *COSTLy* language; scripting by example; inductive learning; assessment integration



**Citation:** Chatzimichalis, G.; Papasalouros, A. Integrating Assessment in a CSCL Macro-Script Authoring Platform. *Appl. Sci.* **2023**, *13*, 1537. <https://doi.org/10.3390/app13031537>

Academic Editor: Arcangelo Castiglione

Received: 30 December 2022

Revised: 18 January 2023

Accepted: 21 January 2023

Published: 24 January 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

“The holy quest of CSCL is to establish environments that directly or indirectly favor the emergence of rich interactions” [1] within groups of collaborating individuals aiming to achieve a non-trivial goal. These intended rich interactions will multiply the efforts and the potential of each member; cultivate or promote required skills; increase inventiveness and productivity; and, in this way, may also scaffold outstanding group achievements [2]. For such a collaboration to be more fruitful, participants should share “their ideas and support them with reasons, discuss different views and resolve these to achieve group consensus” [3].

Numerous researchers adopt the CSCL macro scripts as the path to this holy quest. These scripts are defined by designers (educators or instructional designers) intending to promote collaborative learning by forming the way in which learners act together within a working group. “In specifying a sequence of learning activities, together with appropriate roles for the learners, collaboration scripts are designed to trigger engagement in social and cognitive activities that would otherwise occur rarely or not at all” [4]. The design of such a script should aim to assist teachers to orchestrate, coordinate and manage classrooms collaborative activities [5]. The description of a CSCL script depends upon a set of components—namely, participants, activities, roles, groups and resources—and a set of mechanisms that are group formation, task distribution and activity sequencing [4]. CSCL macro scripts should be structured to scaffold the development of cognitive schemas, so-called internal scripts, about proper activities in the context of a certain social setting [6]. For the rest of this paper, whenever the word “script” is used, it refers to a CSCL macro script.

Moreover, as in any other learning activity, assessment is considered to be a crucial constituent of the plan of each of the above scripts' components and mechanisms, and its configuration should be integrated into the design of the script [7]. An assessment plan is a public document that should summarize the expected assessor's learning results, the methods that are going to be used, the time schedule of assessment phases (e.g., an evaluation test) and the required involvement of the participants. Assessment plans, if designed appropriately, can easily guide students to specific learning aims, monitor learners' progress, evaluate groups' achievements, verify obstacles or misunderstandings and enable teachers to provide the necessary feedback. For example, assessment can be utilized by learners to understand what the main points of a collaborative activity are, what is primarily being asked of them to achieve and to successfully focus their attention on specific goals [8]. They can also use their teacher's assessment feedback to reorganize their work, set new priorities and define a more concrete base for their future efforts. In a CSCL environment, the assessment plan introduces additional design complexity since it should evaluate not only each person's effort but also his/her contribution to the accomplishments of the collaboration group. All the above describes the importance of an appropriate assessment plan during the creation of a CSCL script as well as the complexity of such a task.

Comparisons of individual learning and collaborative learning [9] have demonstrated that the former is more effective if members of a group are committed to the same target, interact productively and their individual achievements can be measured in an objective manner. Collaborative activity is, thus, a demanding environment for learners since it requires them to prioritize aims, schedule the appropriate steps, study proposed resources and finally decide which of these resources are more useful for his/her assignments [10]. Furthermore, in the collaborative group context, any group member should coordinate all of the above with those of the other members of the group. This regulation effort is necessary for smooth and productive cooperation within the group [10]. Probably, these regulation skills are not equally developed for all members of a group. For members with low regulation skills, an appropriate script will remedy this shortcoming. Unfortunately, over-scripting [11] (structuring every aspect of a group collaboration activity) will limit the ability to test and exploit these decisive skills. "Will the fun and the richness of group interactions survive this quest for effectiveness?" [11]. Researchers propose adaptive scripting as a balanced approach to the above question [10,12].

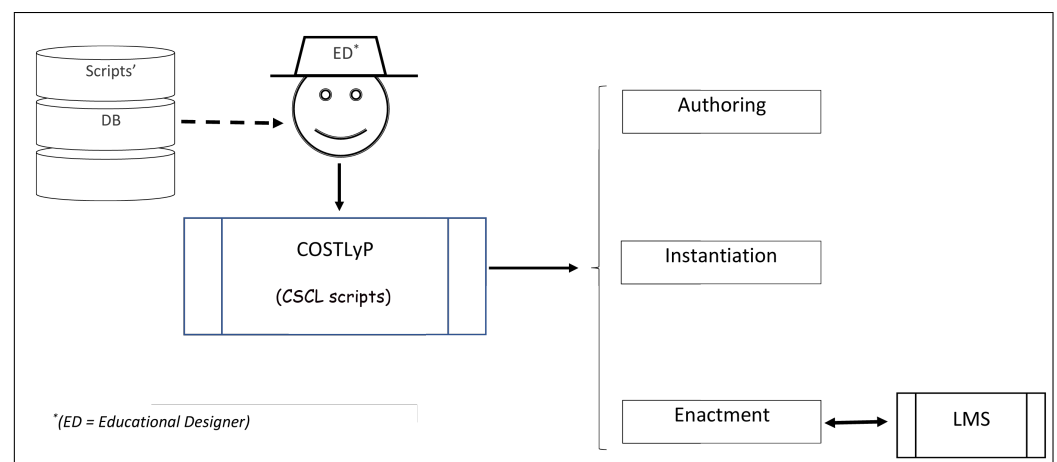
As proposed by IMS-LD (IMS-LD Consortium 2003 [13]), the life cycle of a collaboration script comprises four different phases: 1—Design; 2—Instantiation; 3—Enactment; 4—Evaluation. Although this definition is more than 20 years old, there is no commonly accepted CSCL platform to accommodate all of these phases. This clearly denotes the difficulty of the task, taking into consideration all the aforementioned parameters. Moreover, some of the proposed platforms for the definition of the design phase of a script may require from their users an extensive IT knowledge (e.g., complex XML definition of scripts) or they offer a limited set of predefined patterns. These characteristics may fuddle instructional designers or teachers to author their own scripts or adapt others' scripts to the needs of a specific classroom. Other tools try to facilitate users by providing graphical user interfaces to their functionality.

For example, *Web Collage* [8] is a software IMS-LD authoring tool extending the ideas of its predecessor tool named "Collage" [14] with the integration of assessments capabilities. As its predecessor, it relies on predefined Collaborative Learning Flow Patterns (CLFPs) to synthesize new collaborative activities by combining well-known pedagogical practices. These predefined patterns will enable novice designers to exploit the knowledge and experience of the experts who designed those patterns. The idea of predefined patterns is also followed for assessment design patterns offering a useful amount of expertise using eminent assessment procedures.

In this paper, we will present the assessment-related aspects of COSTLyP, which is a platform that aims to overcome some of the difficulties one meets when dealing with

CSCL script development [15]. The original research question for the creation of COSTLyP is to provide useful and meaningful means for educational designers or teachers to author and deploy CSCL scripts with assessment provisions in a formal manner. This manner should encode CSCL scripts in a consistent and concise way and provide a methodology to facilitate scripts' integration in the everyday reality of an actual classroom leading up to their enactment. A comfortable and concrete solution to the above, will enable more educational practitioners to be involved in the script authoring and deployment process, also taking into account the important constituent of assessment.

We claim that, initially, COSTLyP's usage requires no specialized knowledge of IT and, as preliminary evaluation tests show, scripts' authors and teachers may use it after a small introductory phase. It deals with the first three phases of a CSCL script's life cycle (see Figure 1).



**Figure 1.** Data flow diagram of the work of a user within COSTLyP.

The rest of this paper is structured as follows: the following section briefly describes the COSTLyP platform. In Section 3, the implementation of assessment-related aspects of COSTLyP are discussed as means for authoring adaptable CSCL scripts aiming to, among others, cultivate or promote member's regulation skills. In the last section, conclusions and plans for future work are presented.

## 2. Short Presentation of COSTLyP

COSTLyP is a web-based platform that aims to offer educational practitioners a convenient way to author scripts, to save them in a computer convenient manner, to instantiate them with specific settings (e.g., pupils within a classroom) and finally guide them in the enactment of the selected script. In [15,16], the reader can find a detailed description of the methodology followed for the creation of COSTLyP, an extended presentation of the platform itself and reports on its expressiveness and usability.

For the rest of this paper, we utilize the term “user of the platform” to refer to educational designers, teachers or educational practitioners. In the following subsections, a short description of COSTLyP is presented.

### 2.1. COSTLy Language

COSTLyP implementation starts with the definition of COSTLy [16], a specialized language capable enough to define CSCL scripts and is primarily based on Mathematical Logic. Scripts can be expressed in a logic-based representation as constraints of First-Order Predicate Logic. Common Mathematical Logic syntax can be used in COSTLy with common logical connectives (conjunction, disjunction, negation, implication and equivalence) alongside with the universal and existential quantifiers for sets. Thus, in COSTLy, one can compose adequately complicated and expressive constraints to describe scripts' rules.

COSTLy language defines a *script* as a *set of phases*. In each phase, participants should be allocated to a unique group in the context of a collaborative activity. Such an allocation specifies a *partition*. The union of all the groups of a partition results the set of the participants. Each partition (e.g., group assignment for participants, probably alongside their roles and related resources), in the context of COSTLy, is described as a composite logic constraint. In other words, the script's author should define logical predicates that all the groups within a partition must validate. Of course, several group allocations may validate the same predicate. COSTLyP, in a later phase, will seek and present one of them as a plausible solution for the enactment phase of the script.

In order to give an intuitive example of the above, let us write the second (JG) phase of the popular Jigsaw script [4]. Let  $S$  be the set of all the participant with a cardinality  $|S|$  and a set of books named *books* (let  $|books|$  denote its cardinality). Our specific Jigsaw example, in its first *Expert Group* (EG) phase of the script, allocated participants in  $|books|$  groups, and each group had to study and summarize a distinct book. The information of which book has been summarized by whom is asserted as new knowledge in COSTLyP during the EG phase and can be accessed during later phases. In the second, *Jigsaw Group* (JG) phase of the Jigsaw, members of groups should have read different books during the EG phase.

The COSTLy definition for this JG phase is depicted below (Listing 1).

**Listing 1.** COSTLy definition for the JG phase of the JigSaw script.

```
phase JG:
  create-partition Pj for S, Books with |S| / |Books| groups
  forall Gr in Pj
    forall Book in Books distribute St in Gr
      EG.summarize(St, Book)).
```

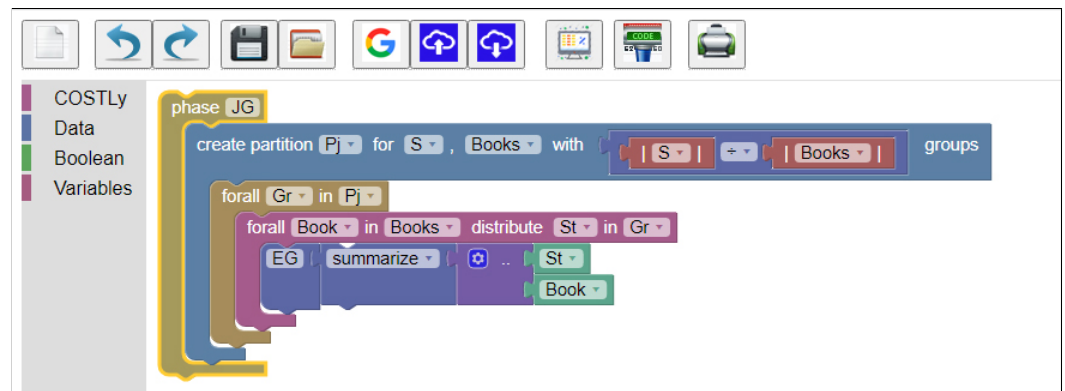
In the above COSTLy code,  $P_j$  is the name for the sets of all groups (partition) and *forall* expression can define predicates over sets. For example, “Forall Gr in  $P_j$ ” expression defines a universal quantification for all groups “Gr” within partition “ $P_j$ ”. Gr is a variable denoting a group of participants.

*EG.summarize(St, Book)* refers to the abovementioned asserted knowledge that relates students with the book that he/she previously summarized in EG phase.

The “distribute” expression is a logical construct of COSTLy and defines a relation among the elements of the two quantified sets (Books and Gr) so that the elements of the second (Gr) are evenly allocated to the elements of the first (Books).

## 2.2. COSTLy Editor

In order to facilitate the learning process of COSTLy language for educational practitioners, we developed an online graphical editor using Google Blockly library [17]. This library enabled us to build an editor such as Scratch [18], transforming the process of writing the textual representation of a script (as in Listing 1) to a puzzle creation in which interconnection of appropriate blocks builds a visual representation of the script, as shown in Figure 2 below.



**Figure 2.** Visual representation of the JG phase of Jigsaw script in editor.

This editor, following its user actions to connect the building blocks of a script, automatically constructs the relevant textual representation of a script in COSTLy language, such as the code in Listing 1.

As depicted in Figure 2, this editor also provides the basic functionality for creating new scripts, for saving them locally or in the cloud, for opening and using stored scripts or for printing them.

The key idea behind this kind of editor is that a big part of the syntax of a language is encoded within blocks and on the allowed interconnections among them. Hence, new users can more easily adapt themselves to script authoring by providing only the necessary blocks' parameters.

### 2.3. Instantiation and Enactment Steps in COSTLyP

The user of COSTLyP can proceed to the second phase of a script's life cycle. It can instantiate the script by providing specific classroom information loaded in the platform from CSV files.

The table in Figure 3 depicts not only the participants in a collaborative activity but also enables the educational practitioner to impose additional restrictions to group formation. He/she can select obligatory allocation of pupils in the same group (such as James and Mark) or forbid others (such as Jennifer and Mark, who cannot be allocated in the same group). In this way, the practitioner may optimally describe the status of the classroom exploiting his/her valuable experience.

**Partition : Pj**Load participants from file : [upload 'parts2.csv'](#)

[James', 'Jennifer', 'Nancy', 'Mark', 'Edward', 'Dorothy', 'Elizabeth', 'David', 'Steven', 'Michael', 'Maria', 'Helen'];

	James	Jennifer	Nancy	Mark	Edward	Dorothy	Elizabeth	David	Steven	Michael	Maria	Helen
James		<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Jennifer			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Nancy				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Mark					<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Edward						<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Dorothy							<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Elizabeth								<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
David									<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Steven										<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Michael											<input type="checkbox"/>	<input type="checkbox"/>
Maria												<input type="checkbox"/>
Helen												

Load tasks from file : [upload 'tasks2.csv'](#) ['Book1', 'Book2', 'Book3', 'Book4'];**Figure 3.** Instantiating JG phase of Jigsaw script.

After the instantiation of a script, COSTLyP can direct all the given information to its dedicated component (backend) that will try automatically to search for a group formation valid in the terms of the set of rules described by the script and the user-determined instance. The results of this process are presented to the end user of COSTLyP. These results can be used for the enactment phase of the script in the user's specific setting. For the JigSaw example, results may have the following form (Figure 4):

Groups : Script : JIGSAW.	
Phase : EG.	Phase : JG.
partition_name : P	partition_name : PJ
Group 1	Michael
	Maria
	Helen
Group 2	Elizabeth
	David
	Steven
Group 3	Mark
	Edward
	Dorothy
Group 4	James
	Jennifer
	Nancy

**Figure 4.** Jigsaw information group allocation for the script's enactment.

The enactment of a CSCL script shapes the succession of activities that must be fulfilled by the specified participants (e.g., in a classroom) of the script as well as the tools and documents that can be used in each activity [19].

#### 2.4. Writing Scripts out of Examples Using Inductive Learning

Aiming for a smoother adaptation to COSTLyP for novice users, we designed an additional tool to guide and help end users in creating their scripts [20]. With this tool, the end user of COSTLyP may describe in a tabular form an example that depicts his/her favorite group allocation, such as the following (Figure 5):

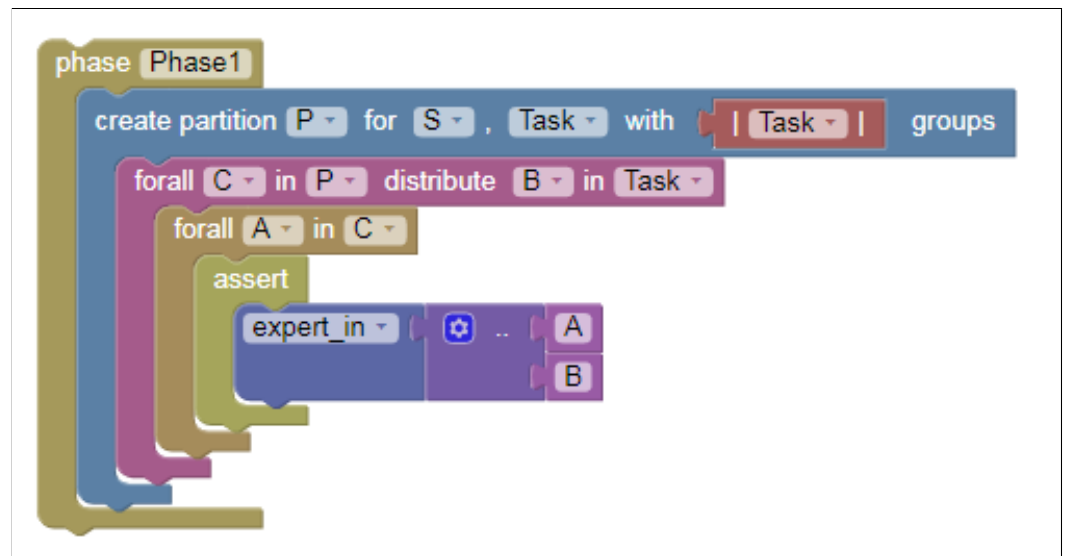
Groups						
	Names	Feature	G1	G2	G3	G4
<input type="checkbox"/>	amelia	1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	thomas	1	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	jack	2	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	emily	1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	ava	1	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	george	2	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	john	2	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	jacob	1	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	olivia	1	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
<input type="checkbox"/>	michael	2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<input type="checkbox"/>	maria	2	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
<input type="checkbox"/>	helen	1	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
			3	3	3	3

**Figure 5.** User tabular example with 12 pupils constituting 4 groups ( $G_1, G_2, G_3, G_4$ ) of 3 participants each.

This tool will analyze the input table and, by the use of inductive learning, will seek for hidden or encoded relationships in table rows [15]. Rows and columns of the given table will be used as positive examples in the relevant induction process. Discovered relationships (if any) will be automatically converted by this tool to COSTLy code, expressing the corresponding general logic constraint, of which user input is an example. To be more specific, mathematical logic is used not only to represent the existing knowledge encoded in the user's tabular example but also to derive new knowledge discovered using logical inference [21].

In our example, the analysis of the given tabular example identified a 1-1 relationship among groups/columns  $\{G_1, G_2, G_3, G_4\}$  and the values in predicate/column *expert\_in* of the input data. Thus, COSTLyP can easily encode this relationship in a COSTLy phase given by the following graphical representation (Figure 6):





**Figure 6.** Visual script description inducted automatically by the example of Figure 5.

### 2.5. Enactment of Scripts

The enactment of a script will be accomplished in cooperation with an LMS (Learning Management System). The basic design of COSTLyP, as a web page, facilitates its interaction with an appropriate LMS using its Application Programming Interface (API). For example, in Moodle's [22] API documentation, one can find its definition for groups and groupings. "Moodle Groups are a way of expressing collections of users within a course" while "groups may be grouped together into named Groupings" [22]. There is a significant similarity between *Moodle Groupings* and the *partition* term of COSTLy language.

Most of the available LMSs are equipped with rich APIs mostly in PHP language (e.g., [23]), offering the ability for external platforms to interact with their internal database programmatically, apart from the standard way by using their user interface. By calling API methods, one can create courses, activities, groups, groupings, resources for groups (e.g., files, URLs), etc. Thus, it is a technical issue to exchange information between COSTLyP and an LMS's database, enriching this database with the processes and mechanisms prescribed within a CSCL script as it is authored and instantiated by a teacher. For example, it is straightforward to populate a script's instantiation table (see Figure 3) using an API call to obtain the list of students enrolled in a specific course defined in an LMS.

Additionally, some of the LMSs offer the ability to programmatically handle events happening inside their user interface. In this way, additional script's sequencing aspects may be handled, e.g., sending a new resource when an assessment plan has been completed, either to a user or to a group, or creating and announcing a new partition of the participants, such as the one created in Figure 4.

From all the above, we can claim that it is quite simple to integrate the interaction with existing LMSs to COSTLyP, handling and controlling, in such a way, the enactment phase of a script.

## 3. Assessment Integration in CSCL Scripts

### 3.1. Methodology

It is obvious that script-integrated assessment results should affect the evolution of a CSCL macro script's enactment phase. Depending on the outcome of an assessment activity, teachers and students may have to reorganize their work, their goals or their research methodology. Thus, sequential execution of a script's phases may be proved insufficient.

To meet the above requirements, scripts should be equipped with adaptation capabilities to focus on the needs of students who participate in a specific collaborative activity. Moreover, this adaptation could be expanded to the extent of the regulation or guidance that is required within each group.



A test for the assessment-related capabilities of COSTLyP could be considered the potential of the platform to express well-known scripts transformed in a way to have the sequencing of their phases to be determined by the results of intermediate assessment.

### 3.2. Assessment Directives in COSTLy Language

In order to integrate some of these aspects, in COSTLyP [15], we include in the grammar definition of COSTLy language a specialized rule that describes the implementation of an assessment activity during the evolution of a script (Listing 2):

**Listing 2.** Assessing rule example.

```
group_results=assess(resource)
```

In the abovementioned rule, the resource can be a URL to the predicted assessment plan's parts (a document, for example), an online questionnaire or whatever an educational practitioner would like to use in the specific CSCL settings. This URL should be determined during the instantiation of this script. The outcome of this process for each group is a set containing, for example, the grade of each member of a group. In our example, the set is stored in a script vector variable named 'group\_results'. This assessment may be performed after the execution of a script's phase. In this way, the outcome of the assessment may verify the progress the participants made after the completion of the phase; their need for teacher's feedback; their accomplishments; and, finally, if they are capable to proceed to the next phase of the script. Additionally, assessment results may point out which should be the next steps for the group or for some members of it.

### 3.3. Conditional Execution of Script Phases

We additionally enrich COSTLy grammar with the capability of the conditional execution of phases in the context of the known "if-else" programming construct, such as in the following two examples (Listings 3 and 4):

**Listing 3.** Conditional execution of phases B, C.

```
phase A
if (condition)
  phase B
else
  phase C
```

**Listing 4.** Repeat phase A, if necessary.

```
phaseA
group_results=assess(resource1_url)
if (condition)
  phaseB
else
  repeat_phase(phaseA, resource2_url)
```

The example of the grammar construct *repeat\_phase(phaseA)* obviously denotes that if the condition is unmet, the group should repeat the *phaseA* of the script.

The "condition" refers to any valid COSTLy expression [15], which is evaluated to a boolean value. For example, a user can combine Listings 2 and 4 as follows (Listing 5):

**Listing 5.** Repeat phase A, if necessary after assessment results.

```
phaseA
group_results=assess(resource1_url)
if (min(group_result)>5)
```

```

    phaseB
else
    repeat_phase(phaseA, resource2_url)

```

The above example may denote that a group can proceed to phase B of the script only if the minimum grade achieved by some members of the group is greater than 5. Otherwise, the specific group should repeat phase A, taking into account the resource provided that may include the teacher's feedback, additional study material, etc. This resource also has to be specified by the teacher during the instantiation of the script.

There are numerous proposals for assessment plans in the educational research field [24–26]. Some of them focus on the need for assessing a person's progress and achievements, while others, in the context of a collaborative activity, try to verify group performance, successes and failures along with the individual contributions to them. Especially, in a collaborative activity, an appropriate strategy for assessment will also motivate pupils to being profoundly more involved in the specific learning process [26].

Our approach for CSCL scripts' authoring does not propose any specific assessment plan to the users. It relies on the teacher's experience to select whatever suits the needs of a specific enactment paradigm of a script. This selection will be made at the instantiation phase of the script's life cycle.

In recent years, due to social circumstances and needs, several tools have been developed to simplify the production of educational material. An example of this is *h5p* [27], by "which authors may create and edit interactive videos, presentations, games, advertisements and more." [27]. H5p users can easily import, export or share their creations. Several researchers have investigated the impact of using these techniques in active peer learning or in collaborative activities [28,29]. Exported objects can easily be connected to COSTLyP, offering rich educational material with the integrated assessment's activities.

### 3.4. Conditional Jigsaw Example with Assessment

Implementing all the above in COSTLyP, users are capable of authoring more complicated scripts. For example, below, a conditional version of the known Jigsaw script is presented in the visual form within the COSTLyP editor (Figure 7).

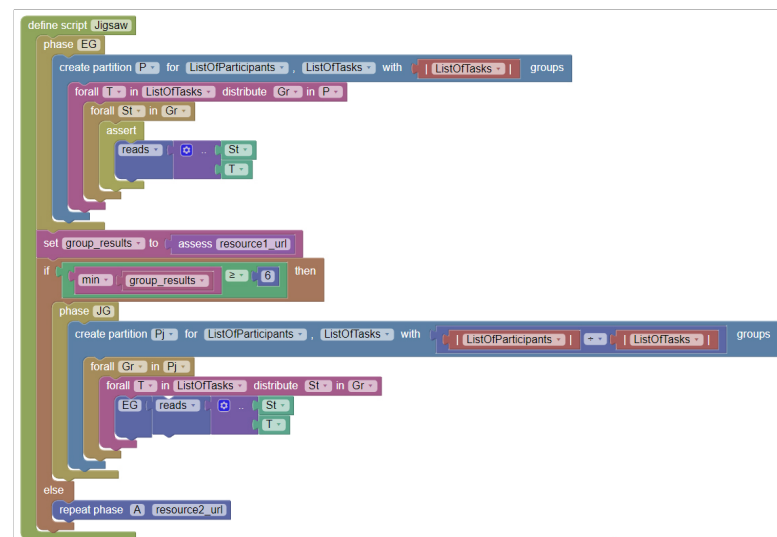


Figure 7. Conditional jigsaw script in web editor of COSTLyP.

## 4. Conclusions and Future Work

The representation of a CSCL script using COSTLy may offer a lot of advantages in the context of the effort to write more adaptive scripts. The implementation of conditional execution of the script's phases, such as the Jigsaw script version shown in Figure 7, clearly provides a new idea, a methodology in which assessment results can influence

the time evolution of the script itself. COSTLyP can assist an educational designer or a teacher to construct scripts that will take into account the assessed results of a collaborative activity versus their expected goals. In this way, when teachers enact those scripts in their classrooms, they can have a significant coordinating equipment that will allow them to adjust their feedback to whole groups, or to specific members of a group or to assess each individual's efforts.

We are aware that some programming structures may hinder the acceptance of this approach from the educators. We believe that the visual editor of COSTLyP and its capability to author scripts by example are some important steps to anticipate some of those hesitations. Additionally, educators may start their avocation with COSTLyP by using libraries of scripts and by instantiating them for their needs. These libraries may be populated by educational designers or by other teachers. This avocation, soon, will enable them to author their own scripts, instead of using other platforms wherein some offer a limited number of predefined patterns and their combinations or some others require advanced expertise in IT.

Our immediate plans involve the experimental formation of a test group of actual teachers who will initially use COSTLyP to instantiate existing scripts within some real educational setting using an online library of CSCL scripts. This library should comprise scripts with assessment activities and conditional execution of the subsequent phases. After this initial phase, and after a short training period, we will ask them to make an effort to author orally-described scripts in COSTLyP and their original scripts. Finally, we plan to ask them to enact them in their classrooms while gathering all the valuable feedback from this process.

Our future work will also focus on the integration of COSTLyP with existing LMSs such as Moodle [22]. Having realized this necessary integration, we can perform additional adjustments and improvements exploiting real educational data (e.g., classrooms, assignments, assessments, etc.) from online and offline collaboration examples. The ultimate goal is to build a useful system, for all who would like to author CSCL scripts, for integrating their assessment plan in those scripts.

**Author Contributions:** methodology, G.C., A.P.; software, G.C., A.P.; formal analysis, A.P.; investigation, G.C.; resources, A.P.; supervision, A.P.; project administration, G.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data sharing not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Dillenbourg, P.; Hong, F. The mechanics of CSCL macro scripts. *Int. J.-Comput.-Support. Collab. Learn.* **2008**, *3*, 5–23.
2. Sanz-Martínez, L.; Martínez-Monés, A.; Bote-Lorenzo, M.L.; Muñoz-Cristóbal, J.A.; Dimitriadis, Y. Automatic group formation in a MOOC based on students' activity criteria. In Proceedings of the European Conference on Technology Enhanced Learning, Toulouse, France, 12–16 September 2022; Springer: Cham, Switzerland, 2017; pp. 179–193.
3. Schwarz, B.B.; Swidan, O.; Prusak, N.; Palatnik, A. Collaborative learning in mathematics classrooms: Can teachers understand progress of concurrent collaborating groups? *Comput. Educ.* **2021**, *165*, 104151.
4. Kobbe, L.; Weinberger, A.; Dillenbourg, P.; Harrer, A.; Hämmäläinen, R.; Häkkinen, P.; Fischer, F. Specifying computer-supported collaboration scripts. *Int. J.-Comput.-Support. Collab. Learn.* **2007**, *2*, 211–224. <https://doi.org/10.1007/s11412-007-9014-4>.
5. Amarasinghe, I.; Hernández-Leo, D.; Ulrich Hoppe, H. Deconstructing orchestration load: Comparing teacher support through mirroring and guiding. *Int. J.-Comput.-Support. Collab. Learn.* **2021**, *16*, 307–338.
6. Vogel, F.; Kollar, I.; Fischer, F.; Reiss, K.; Ufer, S. Adaptable scaffolding of mathematical argumentation skills: The role of self-regulation when scaffolded with CSCL scripts and heuristic worked examples. *Int. J.-Comput.-Support. Collab. Learn.* **2022**, 1–26. <https://doi.org/10.1007/s11412-022-09363-z>

7. Villasclaras-Fernández, E.D.; Hernández-Leo, D.; Asensio-Pérez, J.I.; Dimitriadis, Y.; Martínez-Monés, A. Towards embedding assessment in CSCL scripts through selection and assembly of learning and assessment patterns. In Proceedings of the 9th International Conference on Computer Supported Collaborative Learning, Rhodes, Greece, 8–13 June 2009.
8. Villasclaras-Fernández, E.; Hernández-Leo, D.; Asensio-Pérez, J.I.; Dimitriadis, Y. Web Collage: An implementation of support for assessment design in CSCL macro-scripts. *Comput. Educ.* **2013**, *67*, 79–97.
9. Azevedo, R. Understanding the complex nature of self-regulatory processes in learning with computer-based learning environments: An introduction. *Metacognition Learn.* **2007**, *2*, 57–65.
10. Wang, X.; Kollar, I.; Stegmann, K. Adaptable scripting to foster regulation processes and skills in computer-supported collaborative learning. *Int. J.-Comput.-Support. Collab. Learn.* **2017**, *12*, 153–172.
11. Dillenbourg, P. Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In *Three Worlds of CSCL. Can We Support CSCL*; Open Universiteit Nederla: Heerlen, The Netherlands, 2002; Volume 61491.
12. Walker, E.; Rummel, N.; Koedinger, K.R. CTRL: A research framework for providing adaptive collaborative learning support. *User Model.-User-Adapt. Interact.* **2009**, *19*, 387–431.
13. Consortium, I.G.L.; et al. IMS learning design specification. Retrieved Febr. **2003**, *7*, 2009.
14. Hernández-Leo, D.; Villasclaras-Fernández, E.D.; Asensio-Pérez, J.I.; Dimitriadis, Y.; Jorrín-Abellán, I.M.; Ruiz-Requies, I.; Rubia-Avi, B. COLLAGE: A collaborative Learning Design editor based on patterns. *J. Educ. Technol. Soc.* **2006**, *9*, 58–71.
15. Papasalouros, A.; Chatzimichalis, G. An authoring platform for CSCL script definition. In Proceedings of the International Conference on Human-Computer Interaction, Copenhagen, Denmark, 19–24 July 2020; Springer: Cham, Switzerland, 2020; pp. 625–640.
16. Papasalouros, A. Formalizing CSCL scripts with logic and constraints. In Proceedings of the European Conference on Technology Enhanced Learning, Leeds, UK, 3–5 September 2018; Springer: Cham, Switzerland, 2018; pp. 660–663.
17. Google. Blockly—A Library for Visual Programming Editors. Available online: <https://developers.google.com/blockly/> (accessed on 11 November 2020).
18. Scratch Foundation. Create Stories, Games, and Animations. Share with Others around the World. Available online: <https://scratch.mit.edu/> (accessed on 11 November 2020).
19. Bote-Lorenzo, M.L.; Gómez-Sánchez, E.; Vega-Gorgojo, G.; Dimitriadis, Y.A.; Asensio-Pérez, J.I.; Jorrín-Abellán, I.M. Gridcole: A tailorable grid service based system that supports scripted collaborative learning. *Comput. Educ.* **2008**, *51*, 155–172.
20. Papasalouros, A.; Chatzimichalis, G. Towards CSCL Scripting by Example. In Proceedings of the International Conference on Intelligent Tutoring Systems, Athens, Greece, 8–12 June 2020; Springer: Cham, Switzerland, 2020; pp. 306–315.
21. Flach, P.; Sokol, K. *Simply Logical: Intelligent Reasoning by Example—Online Edition*; John Wiley: Hoboken, NJ, USA, 2018. <https://doi.org/10.5281/zenodo.1156977>.
22. Online Learning, Delivered Your Way. Available online: <https://moodle.com/> (accessed on 15 October 2022).
23. Developer Resource Centre: Learn How to Build, Develop, and Contribute to the World's Most Customisable Learning Management System. Available online: <https://moodledev.io/> (accessed on 15 October 2022).
24. Shepard, L.A. The role of assessment in a learning culture. *Educ. Res.* **2000**, *29*, 4–14.
25. Boud, D.; Cohen, R.; Sampson, J. Peer learning and assessment. *Assess. Eval. High. Educ.* **1999**, *24*, 413–426.
26. Black, P.; Wiliam, D. Assessment and classroom learning. *Assess. Educ. Princ. Policy Pract.* **1998**, *5*, 7–74.
27. Create, Share and Reuse Interactive HTML5 Content in Your Browser. Available online: <https://h5p.org/> (accessed on 18 October 2022).
28. Singleton, R.; Charlton, A. Creating H5P content for active learning. *Pac. J. Technol. Enhanc. Learn.* **2020**, *2*, 13–14.
29. Wehling, J.; Volkenstein, S.; Dazert, S.; Wrobel, C.; van Ackeren, K.; Johannsen, K.; Dombrowski, T. Fast-track flipping: Flipped classroom framework development with open-source H5P interactive tools. *BMC Med. Educ.* **2021**, *21*, 1–10.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.